VOKALEE®

EXPECT THE ANSWER™

WHITE PAPER

The Conversational Voice Infrastructure Layer

Why real-time voice requires more than a large language model — and why regulated voice requires no generative model in the answer path at all.

> *Most conversations do not require artificial intelligence — they require understanding.*



A technical white paper for platform builders, regulated operators, and OEMs.

Executive Summary

Every company building a conversational voice product is rebuilding the same thing: the layer between the phone call and the answer. Speech recognition, voice-activity detection, interruption handling, DTMF, tone detection, call control, audio streaming, latency optimization, telephony-aware scaling. It is the hardest part of the stack and nobody's core product — time-consuming, error-prone, and outside most teams' competency, yet rebuilt from scratch across the entire industry.

Vokalee is that layer — built once, hardened in production, and exposed through a trivial interface. It handles the complete telephony and real-time-audio path and answers from the operator's own authorized information.

What makes it defensible is not another model. It is an architectural decision most of the market got wrong. The prevailing assumption is that every utterance should be routed through a large language model. That assumption is the direct cause of the latency, cost, and — most importantly for regulated operators — the *trust* problems that plague today's voice AI. A probabilistic model can be fluent and confident and still be wrong, and it cannot prove it wasn't.

Vokalee starts from the opposite premise. **A deterministic, microsecond-scale comprehension engine answers from the operator's governed information — and no generative model composes, selects, or authors any answer it speaks.** When the deterministic engine cannot resolve a request, it does not hand the question to a model to guess at. It asks a clarifying question, declines honestly, or routes the caller to a human with the authority to answer. The escalation path is to a person, never to a model.

The result is a system that answers in human conversational time, costs a fraction of API-dependent stacks, runs on instances small enough to be disposable, can run on-premise or air-gapped, and — the property regulated operators cannot get anywhere else — **never manufactures an answer.** This paper explains how it works and why it is the right foundation for voice where being wrong has consequences.

At a Glance

- **No generative model in the answer path.** Every answer is retrieved from authorized information and spoken deterministically. No model composes or selects what a caller is told. When the engine cannot answer, it clarifies, declines, or routes to a human — never to a model.
- **Native DSP signaling** *(a first, as far as we are aware)*. Reads DTMF, ringing, busy, and answering-machine tones from the audio itself — signals a transcript-only platform cannot see.
- **Just-in-time, disposable telephony** *(a first, as far as we are aware)*. Nodes are created for demand and retired after it — power-on to accepting calls in ~1.8 seconds.
- **Provable, not plausible.** Answers are deterministic and reproducible: the same request against the same authorized information yields the same answer, every time — and the operator controls, to the word, what a caller may be told.

- **~18% of a typical stack's cost.** The saving is architectural, so it holds — and widens — as volume grows.
- **Small enough to run anywhere.** A full node fits on a 512 MB ARM instance — the same engine runs in the cloud, on-premise, or inside an air-gapped appliance.

> **A note on terms.** *STT/TTS* — speech-to-text and text-to-speech. *SIP* — the protocol that sets up and controls phone calls. *PSTN* — the ordinary public telephone network. *DTMF* — the touch-tones a keypad makes. *DSP* — analyzing the raw audio signal directly. *VAD* — detecting when someone is actually speaking. *Barge-in* — letting a caller interrupt mid-sentence. *PBX* — a business phone system.

---

1 · The Latency Problem Is an Architecture Problem

The standard conversational-voice pipeline looks like this:

> Speech → STT → **LLM** → TTS → Audio

Each stage adds delay, and the LLM stage is the expensive one — in milliseconds and in dollars. Stacks built this way commonly operate with end-to-end response times from the high hundreds of milliseconds to multiple seconds. That is fine for a demo. It is not fine for a conversation.

Human conversational turn-taking operates on the order of a couple hundred milliseconds. When a system exceeds that, callers interrupt it, talk over it, repeat themselves, and perceive it as slow — regardless of how good the underlying model is. Latency, not model quality, is what makes a voice agent feel artificial.

The reflex in the industry has been to make the models bigger and faster. But the bottleneck is structural. You cannot model your way out of a pipeline that routes a request for store hours through the same multi-hundred-millisecond reasoning path as a complex negotiation. The fix is architectural: **stop sending every utterance to a model.** Vokalee goes one step further for governed voice — it sends *no* answer through a generative model at all.

---

2 · Most Business Conversations Are Predictable

Look at what callers actually say on business lines. The overwhelming majority of interactions are:

- Hours of operation
- Location and directions
- Appointment and order status
- "Press 1 / transfer me to a person"
- A small set of frequently asked questions

These require recognition and retrieval, not reasoning — and in regulated settings, they require being *correct and current*, not merely fluent. Treating them as AI problems is wasteful and, worse, it makes the experience less trustworthy: it pays the full latency and cost penalty of a language model, and accepts its fabrication risk, to answer a question a governed lookup can answer in microseconds with a provable source. This is the insight the rest of the stack is built on.

---

3 · Deterministic Comprehension, and Honest Escalation

Vokalee resolves each turn deterministically, and where it cannot, it escalates to a person — not to a model.

The engine (no AI). The comprehension engine answers most calls without any AI at all. Written in C — chosen for raw speed — it breaks the caller's sentence into words, labels the job each word is doing (nouns, verbs, negations), and from that works out what the caller actually wants: whether they asked a question, and whether they said the opposite of what a keyword suggests ("I do *not* want to cancel"). It then finds the answer in the operator's own authorized information and replies. Labeling a word takes tens of microseconds; deciding a whole response takes about 0.04 ms per turn on a production machine — thousands of times faster than a blink, and far faster than sending the question to a model and waiting. To the caller, the reply is effectively instant.

(One honest clarification, since 0.04 ms sounds too good to be true: that is only the time to decide what to say. What a caller experiences across a full turn is set mostly by two other steps — turning speech into text on the way in, and text back into audio on the way out — which is why caching those locally matters more to felt speed than the engine's raw number. The architectural point holds either way: the step everyone else makes slow and expensive, Vokalee makes nearly free.)

What happens when the engine can't answer — and why this is the whole point for regulated voice. An LLM-first system, faced with a question it cannot ground, generates a plausible answer anyway. That is precisely the failure a bank, a hospital, a school, or an emergency platform cannot tolerate. Vokalee does the opposite. When a request does not resolve to a single authorized answer, it takes one of three deterministic dispositions:

- **Clarify.** If the request is ambiguous — it fits more than one thing the operator governs — Vokalee asks a disambiguating question among the answers it actually holds. It does not guess which one the caller meant; it lets the caller choose. Clarification is a question among known answers, never a re-interpretation.
- **Decline and route.** If there is no authorized answer, Vokalee says so plainly and routes the caller to the person who can answer — it does not fabricate one, and it does not simply take a message. The question is carried to a human with its full context.
- **Hand off the reserved act.** Where communicating the answer is itself a licensed or reserved act — a diagnosis, a legal determination, an emergency instruction — Vokalee connects the

caller to the authorized human and speaks nothing of the answer itself. The act stays with the person entitled to perform it.

> **The regulated guarantee.** No generative model composes, selects, or authors any answer Vokalee speaks, and no model output is ever treated as the operator's authorized answer. Speech-to-text converts a caller's audio into text as untrusted input — it never authors a reply. When Vokalee does not have a governed answer, it clarifies, declines, or routes to a human. It never escalates a question to a model to answer. This is verifiable by inspection of the answer path, not a matter of trust in a vendor's tuning.

The intelligence of the system is not in making every response artificial. It is in answering from authorized fact where it can, and knowing — deterministically — when to stop and bring in a person.

---

4 · The Telephony Layer Nobody Wants to Build

Routing is only half the story. The other half is the real-time audio and telephony work that sits below any voice system, and it is the part AI-first teams consistently underestimate. Vokalee owns all of it.

> **Key idea — The Transcript Fallacy.** Speech-to-text transcripts are not phone calls. A call carries far more than words: tones, interruptions, the signaling beeps that ride inside the audio, whether the line is ringing or busy, emotional cues. A transcript-only system sees only the words — and behaves as though the rest were not there.

A bot builder is not in the DSP business; by design they rely on their STT engine for everything the system "hears." But STT returns words, and words are only part of what happens on a call. Everything else lives in the audio, and Vokalee surfaces it:

- **Native DTMF and tone detection through DSP** — reading the raw audio signal, not a transcript of it. STT providers return words, not touch-tones, so "press 1 for billing," account-number entry, and IVR navigation are out of reach for transcript-only stacks. *Banking, healthcare, and government voice systems are not viable without it.*
- **Call-progress tones** — ringing, busy, fast-busy — so the platform knows what the far end is actually doing.
- **Answering-machine and voicemail detection** — the beep that means *start speaking now* is a complete instruction containing no words, invisible to a transcript. Vokalee knows a machine answered and the exact moment to begin.
- **Answer (off-hook) detection** on outbound calls, at the signaling layer — so Vokalee can respond, and disclose, the instant someone answers, at human timing, instead of the dead-air pause that makes autodialed calls obvious.

- **Voice-activity detection** tuned for real-world phone audio, and **real-time interruption handling** so a caller can talk over the system and be heard immediately.
- **Complete SIP and media management** — answer, transfer, hold, hangup, media bridging — handled at the carrier layer on hardened, production-grade telephony.

None of this appears in a transcript, and none of it is something a bot author can add on their own. DSP-based DTMF and tone detection have run in interactive voice systems for decades; what is new is carrying them into the conversational stack. The current generation of AI-voice platforms was built speech-to-text-first and inherited none of it. As far as we are aware, Vokalee is the only conversational-voice platform that carries the full DSP signaling layer natively — telecom-native in an AI-native field.

---

5 · The Integration Surface

Vokalee abstracts the entire stack above into a single JSON-over-socket protocol. The operator's answer logic receives structured events and transcribed speech, and returns the authorized text to speak plus a small set of control commands. A simplified exchange:

```
Vokalee → logic: { "control": { "new-call": true, "caller-number": "…", "dialed-number":
"…" } }
logic → Vokalee: { "control": { "call": "answer", "language": "en" } }
Vokalee → logic: { … "transcript": "I'd like to book an appointment" … }
logic → Vokalee: "I'd be happy to help with that. What day works best?"
Vokalee → logic: { … "transcript": "[voice:start]" … }   // caller barged in
logic → Vokalee: [interrupt]                               // Vokalee stops TTS instantly
logic → Vokalee: { "control": { "call": "transfer", "destination": "…" } }
```

The operator writes *text in → authorized answer → text out* and inherits low-latency voice, DTMF, barge-in, and call control. Integration is measured in hours, not the weeks or months it takes to build this layer in-house.

Beyond voice: mid-call SMS. Because Vokalee owns the telecom layer rather than renting it, a call and a text message can belong to the same session across two channels. A caller can text Vokalee mid-call and Vokalee can text back — no second messaging vendor, no hand-synced state. This routes around the one thing speech recognition reliably struggles with: alphanumerics. Email addresses, confirmation codes, unusual spellings can arrive as clean text instead of endless "was that P as in Papa?" re-prompts; and in the other direction Vokalee can send a link, an address, or a confirmation the caller can save rather than memorize. *(In testing; a few limitations remain before general availability.)*

---

6 · Just-In-Time, Disposable Infrastructure

Disposable telephony is not a tuning of how telecom is run today — it is a reversal of it. Conventional infrastructure assumes always-on servers and long-lived instances, because

standing a node up is slow, so nodes are kept warm and treated as precious. Vokalee assumes the opposite: instances are created for demand, destroyed after it, and treated as disposable. This is possible because the software is deliberately small — the core engine is C with almost nothing else it needs to run, and an entire server runs comfortably on a minimal ARM machine while carrying many callers at once.

Measured in production vs. typical alternatives · ARM node, June 2026:

	Vokalee	Typical alternative
Power-on to accepting calls	**~1.8 s**	tens of seconds to minutes *(containerized/VM stacks)*
Signaling process ready to take a call	**< 5 ms**	hundreds of ms to seconds *(containerized SIP)*
Deciding a response, per turn	**~0.04 ms**	hundreds of ms to seconds *(LLM inference per turn)*
Memory per concurrent bot	**< 8 MiB**	often hundreds of MB *(a container per session)*

Vokalee figures measured on the node above. Typical ranges reflect containerized/orchestrated telephony and LLM-routed pipelines, not a single named vendor.

Web auto-scaling does not work for voice: web requests are short and interchangeable; calls are long-lived, cannot be interrupted mid-session, and demand changes faster than capacity. Vokalee's scaler is purpose-built — it provisions ahead of demand, schedules new calls onto warm capacity, and retires instances only after their calls drain.

7 · The Economics

For anyone reselling or embedding voice infrastructure, the margin structure is the business case. Vokalee runs on the order of **18% of what a conventional conversational-voice stack costs to operate**, all-in, including speech-to-text and synthesis. That gap comes from avoiding the two cost sinks of API-dependent stacks: Vokalee does not route turns through a paid LLM, and it does not pay telco-API markup on the media path. Synthesis is cached and coalesced; responses are served from local caches rather than round-tripped to remote APIs; idle capacity is minimized by disposable scaling.

Because cost per minute and latency are driven by the *same* architectural decisions, they improve together rather than trading off. The design that makes conversations feel instant is what makes the unit economics work — and because the saving is architectural rather than a discount, it does not erode as volume grows the way per-minute API and LLM fees do.

8 · Proven in Production

This is not a prototype. Vokalee has run in production on real inbound guest calls — answering live, navigating real-world audio conditions, and operating within the cost and latency envelope above. The architecture has been hardened against the things that only show up in production — STT edge cases, transcription timing, overflow, graceful failure — not the clean conditions of a demo.

> **Production deployment.** Live guest calls at a Wyndham property, Orlando / Lake Buena Vista. The platform ran on real customer traffic and was hardened by production usage rather than laboratory conditions.

There is a second layer to this proof that matters as much as the uptime, and it is the one that speaks directly to regulated operators. **This customer did not want an AI solution. They wanted to control, to the word, exactly what a caller is told** — brand voice, accuracy, and consistency ahead of open-ended cleverness. That is precisely what a deterministic architecture delivers and an LLM-first one cannot promise: scripted, predictable, authorized answers where they matter. A real, brand-conscious enterprise chose deterministic control on purpose — which is the same reason a school district, a hospital, or an emergency platform will.

---

9 · From Cloud to Appliance — including air-gapped

The same property that makes Vokalee disposable — a deliberately small footprint running full conversational voice on a minimal ARM node — means it is not confined to the cloud. The deterministic engine, the DSP audio path, and the call-control layer are compact enough to run at the edge, on-premise, or embedded in a dedicated appliance.

For regulated operators this is decisive. Vokalee can be deployed as an on-premise unit behind a PBX, a call-center appliance, an enterprise voice gateway, or inside a **sovereign, air-gapped environment where cloud services are not an option** — the deployment surface a platform that depends on a remote model API structurally cannot reach. One engine serves all of them: Vokalee is designed to be small everywhere, and because no answer is generated by a remote model, no governed answer ever has to leave the operator’s own perimeter.

---

10 · Session-Aware Conversational Infrastructure, with Strict Isolation

Today’s voice systems are amnesiac: when the call ends, everything is forgotten, and every call starts from zero. The next generation preserves continuity — recognizing a returning caller, carrying context forward, resuming later — *while holding strict isolation between tenants.* Vokalee is being designed toward this, and the groundwork is the same discipline that already makes it fast and cheap: the design choices that let one machine carry many callers at once — walling each caller off from the others and using very little memory per caller — are exactly what

it takes to keep every operator’s conversational memory fast and strictly separated from every other’s. The architecture that keeps a single call light is the architecture that keeps a million sessions separate — and separation is a compliance property, not just an efficiency one.

11 · Why This Matters Now

Three trends are colliding: businesses increasingly expect round-the-clock voice; callers expect a competent system at any hour; and existing voice architectures remain too slow, too expensive, and — for anyone accountable for what their system says — too unprovable to meet that demand at scale. The market does not need another model. It needs infrastructure capable of delivering conversational voice at human latency, software economics, and *governed, provable answers*. That gap is open now — before the category settles on the assumption that every call must run through an LLM.

12 · Why Existing Approaches Struggle

Approach	What it leaves unsolved
Transcript-only STT APIs	Cannot observe DTMF, tones, or signaling — only words
LLM-first architectures	High latency and cost on every turn — and no way to prove an answer wasn’t fabricated
Web auto-scalers	Not call-aware; cannot drain long-lived sessions gracefully
Traditional telecom stacks	Expensive always-on idle capacity
Custom in-house builds	Months reinventing commodity infrastructure

Vokalee was designed to address all five at once — which is what makes it infrastructure rather than a point solution.

13 · Why This Is Hard to Replicate

Every investor and partner asks the same question: what stops Twilio, OpenAI, or Amazon from building this? The answer is not any single feature. It is that the necessary pieces do not naturally coexist in one company:

- **Deep telephony expertise** — SIP, carrier quirks, audio handling, the ways real networks fail — two decades of it, the kind that cannot be hired in a quarter.

- **DSP fluency** — reading the signal rather than the transcript — common in legacy telecom, rare in the AI-voice generation.
- **Production-hardened edge cases** — transcription timing, overflow, graceful failure — earned only on real calls.
- **A deterministic comprehension engine** — an architecture the rest of the field's LLM-first assumptions actively work against.
- **Governance built into the answer path** — authorized answers, honest decline, and no generative model between the operator's facts and the caller — which cannot be bolted onto an LLM-first stack, because it is the opposite of how that stack is built.
- **Telephony-aware, disposable scaling** — purpose-built for long-lived calls, not adapted from web auto-scaling.

The AI labs are the telling case. They build extraordinary models, but they have no telephony stack and no governance layer — and their entire product is the generative answer Vokalee deliberately keeps out of the path. An AI-first company would have to acquire telephony depth it lacks *and* re-architect against its own model-first premise. A telecom incumbent would have to rebuild around a disposable, deterministic-first architecture that fights its own always-on, API-metered business model. The hard, unglamorous layer beneath — the one that has to work, and be provable, on every single call — is the part already built, already proven, and already years ahead.

---

14 · Why Vokalee

The future of conversational voice is unlikely to be *everything → LLM*. It is far more likely to be *most requests → answered deterministically from authorized fact, and the rest → escalated honestly to a human.* The companies that win regulated voice will not have larger models; they will have infrastructure designed for the timing, the economics, and the **accountability** of human conversation. Vokalee is that infrastructure.

- **Architectural, not incremental** — differentiation from hierarchical deterministic processing and disposable infrastructure, not a marginally better model.
- **No generative model in the answer path** — the one assurance a regulated operator cannot get from an LLM-first platform.
- **Telecom-first** — DSP-native DTMF and tone detection that transcript-only pipelines cannot provide — the unlock for banking, healthcare, government, and emergency voice.
- **Conversational latency in human time** — deterministic processing at ~0.04 ms per turn, with STT and synthesis kept tight through local caching.
- **~18% of a typical stack's operating cost** — room to price below incumbents and hold healthy margins.
- **A trivial integration surface** — text in, text out — connected in hours.
- **Proven in production on real calls**, by an enterprise that chose deterministic control on purpose.

- **Small enough to run anywhere** — from disposable cloud instances to an on-premise or air-gapped appliance.

Next Step: A Technical Evaluation

The fastest way to assess Vokalee is to put a real call through it. A pilot is scoped to be low-effort:

1. Connect your authorized answer logic to the Vokalee protocol — a socket, transcribed text in, your response text out. Typically a few hours.
2. Run live calls through a provisioned node and measure what matters: end-to-end conversational latency, DTMF handling, barge-in behavior, honest-escalation behavior, and cost per minute against your current stack.
3. Review results together and model the economics at your projected volume.

To schedule a technical evaluation and pilot, contact the Vokalee team.

Vokalee® is a registered trademark. This document contains forward-looking statements about planned capabilities; roadmap items are identified as such.